AFOSR Program Review

Formal Specification and Design of Secure Agents (99NI029)

Agent Development Environments for Large-Scale Multi-Agent, Distributed Mission Planning and Execution in Complex Dynamic Environments (99NM097)

Scott A. DeLoach

Thomas C. Hartrum

March 2000

The views expressed in this presentation are those of the author and do not reflect the official policy of the United Stated Air Force, Department of Defense, or the US Government.

Approved for Public Release; Distribution Unlimited.

Overview

- Problem statement, objective
- Current work, results
- Significant accomplishments/transitions
- Future direction
- Publications
  - Significant publications and total publications under the grant.
  - Also mentions in the media.
- Issues

Problem Statement

Apply software synthesis techniques to the analysis, design, and construction of intelligent agents to ensure appropriate security and communications protocols are correctly incorporated.

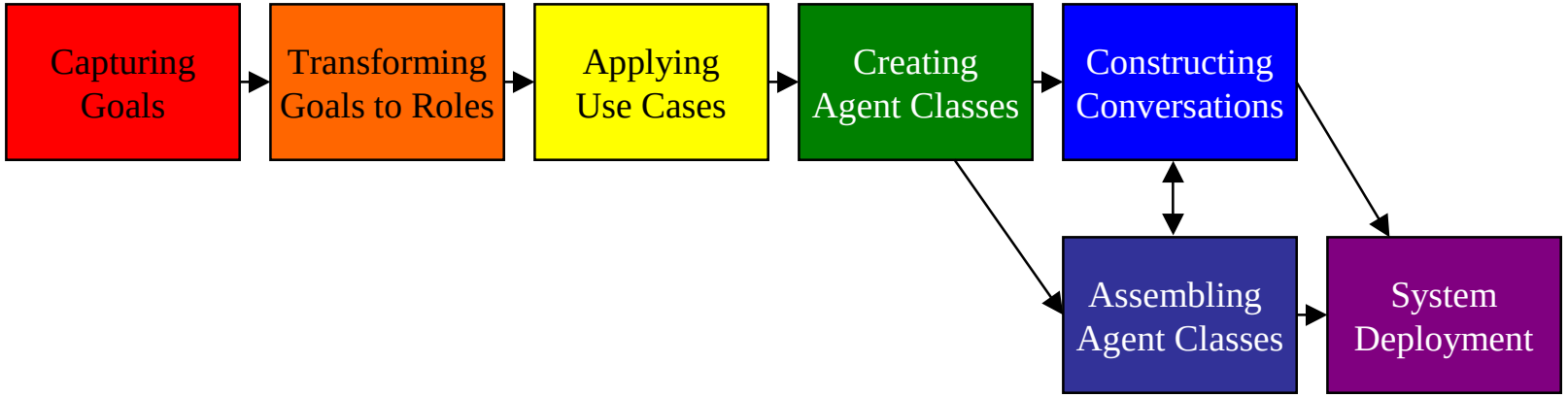
- Define a methodology and language for defining high-level behavioral specifications for multiagent systems.
- Implement a system to design and semi-automatically synthesize multiagent systems that adhere to required protocols.
- Verify that systems meet requirements.

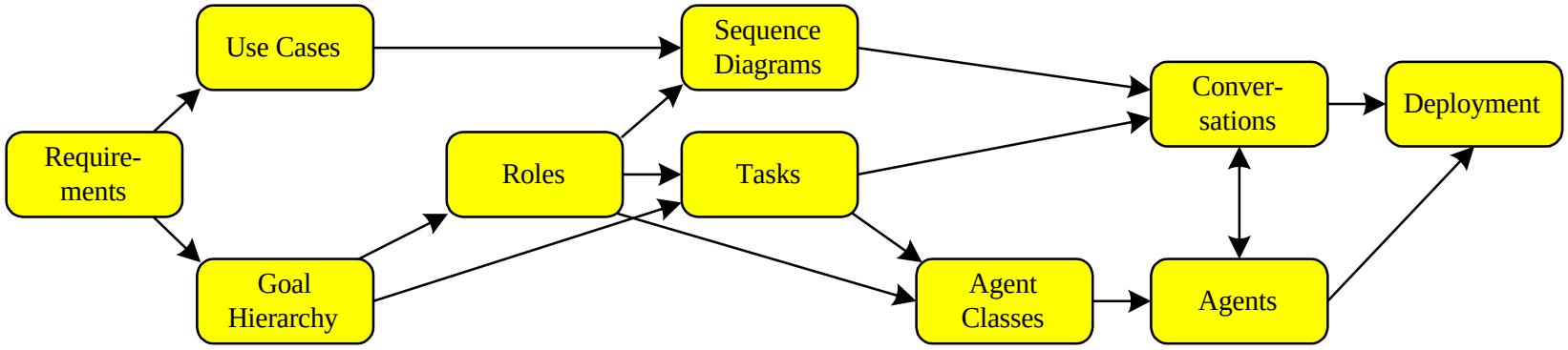
Current Results

- Multiagent Systems Engineering (MaSE)
  - Specification to code methodology for building multiagent systems
- agentTool
  - Automation for MaSE
  - Supports design, verification, and code generation
- AWSOME

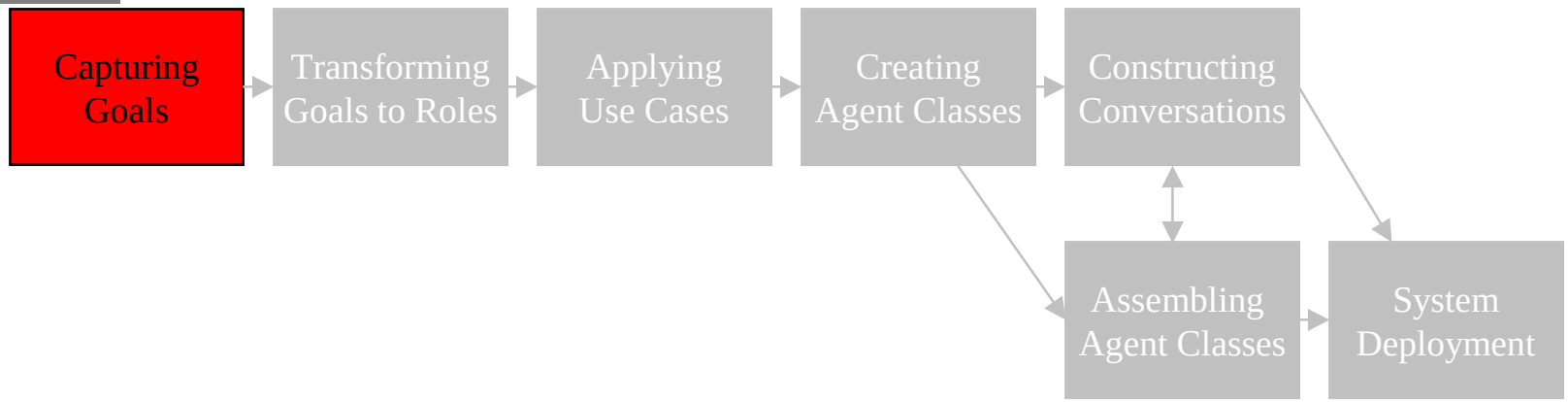
Multiagent Systems Engineering

Multiagent Systems Engineering (MaSE)

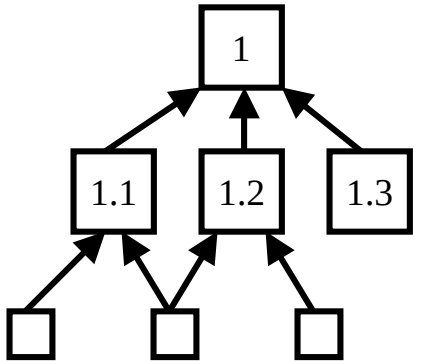




Capturing Goals

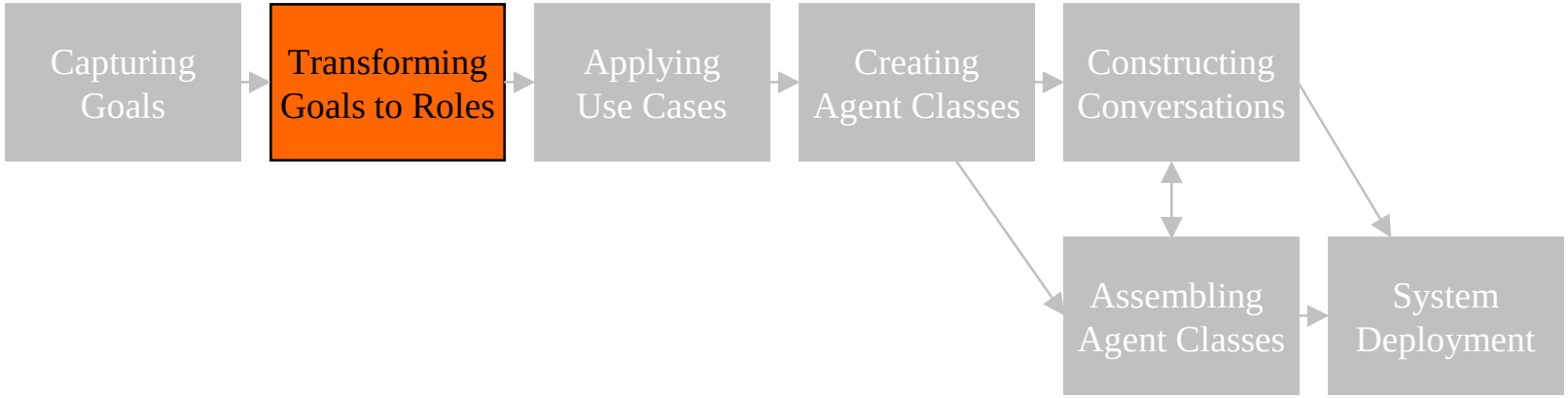


- Input
  - System specification / requirements
- Product
  - A structured hierarchy of goals
  - Use cases
- Diagrams
  - Goal hierarchy diagram

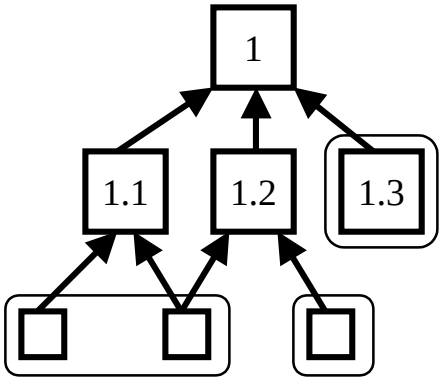


Goal hierarchy diagram (simplified)

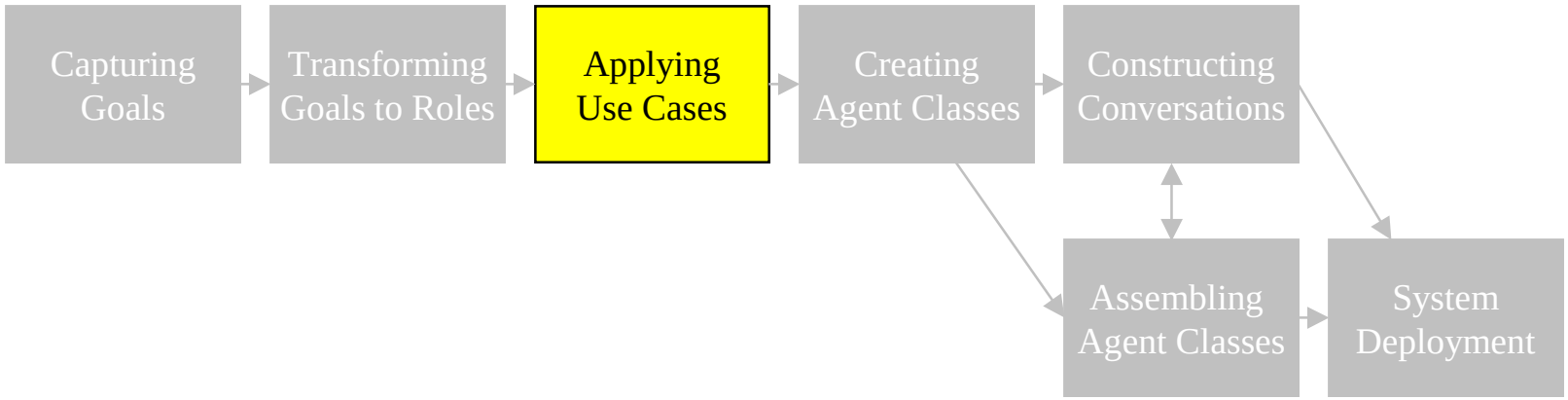
Transforming Goals to Roles



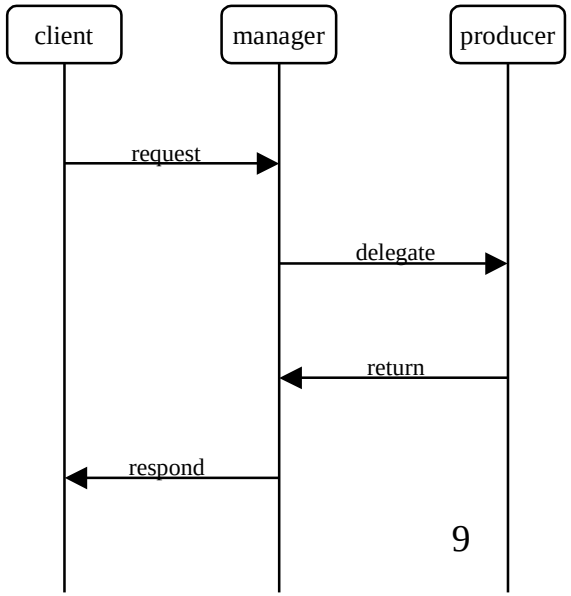
- Input
  - Goal Hierarchy Diagram
- Product
  - Agent Roles
- Diagrams
  - Role Models



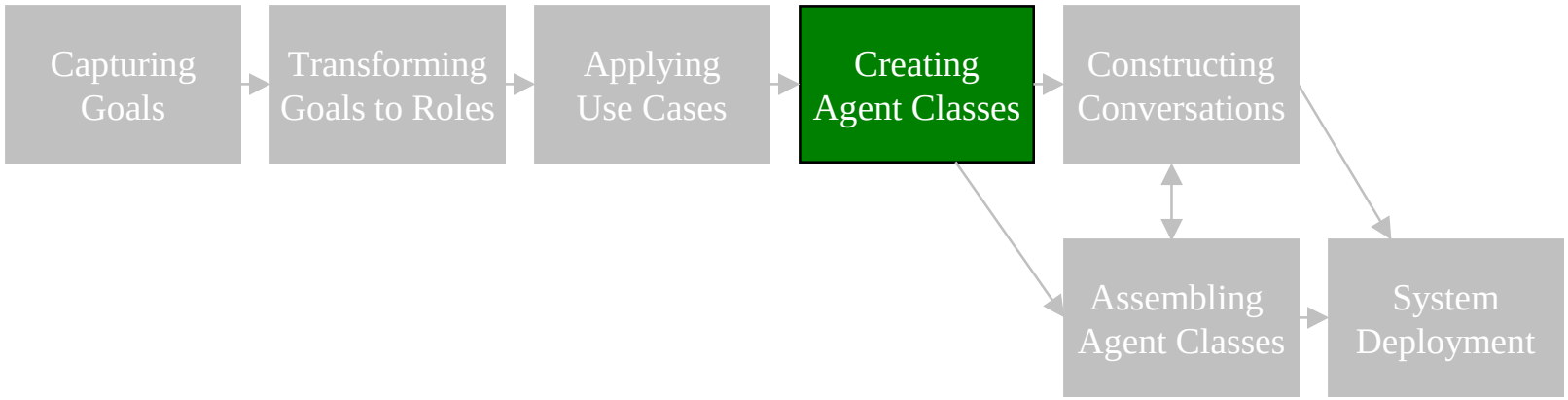
Applying Use Cases



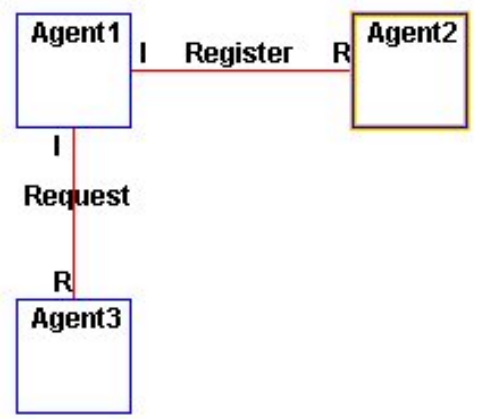
- Input
  - Use cases
  - Set of roles
- Product
  - Minimum communication paths between roles
- Diagrams
  - Sequence Diagram
  - Task Diagram



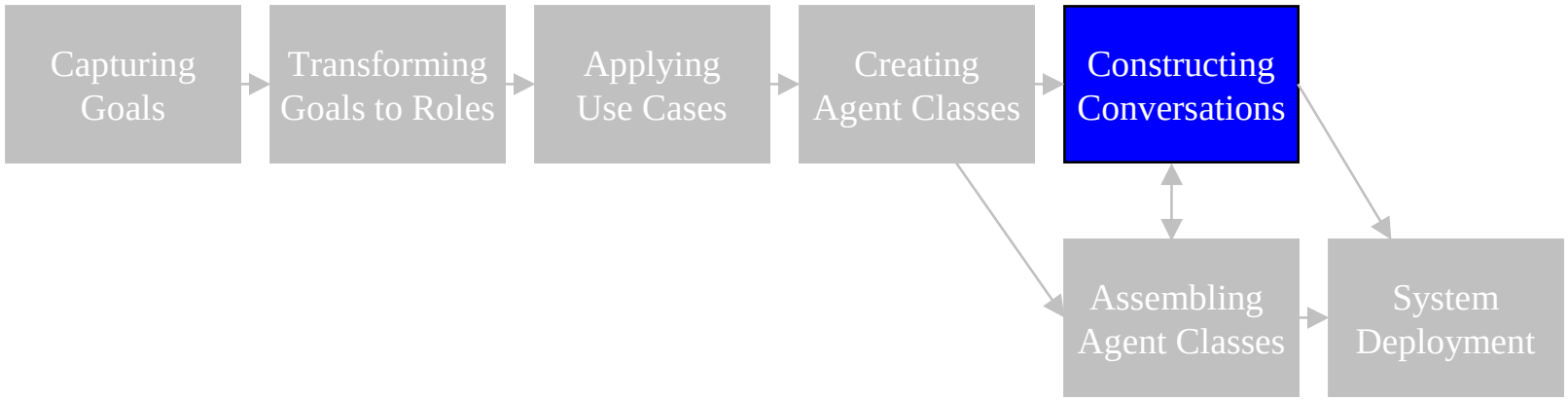
Creating Agent Classes



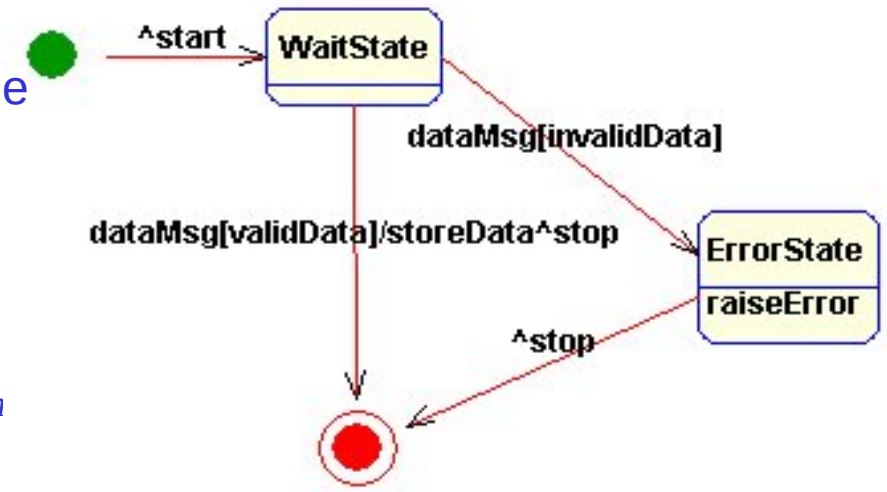
- Input
  - Agent roles
  - Tasks
- Product
  - A diagram of agent classes and conversations
- Diagrams
  - Agent Class Diagram



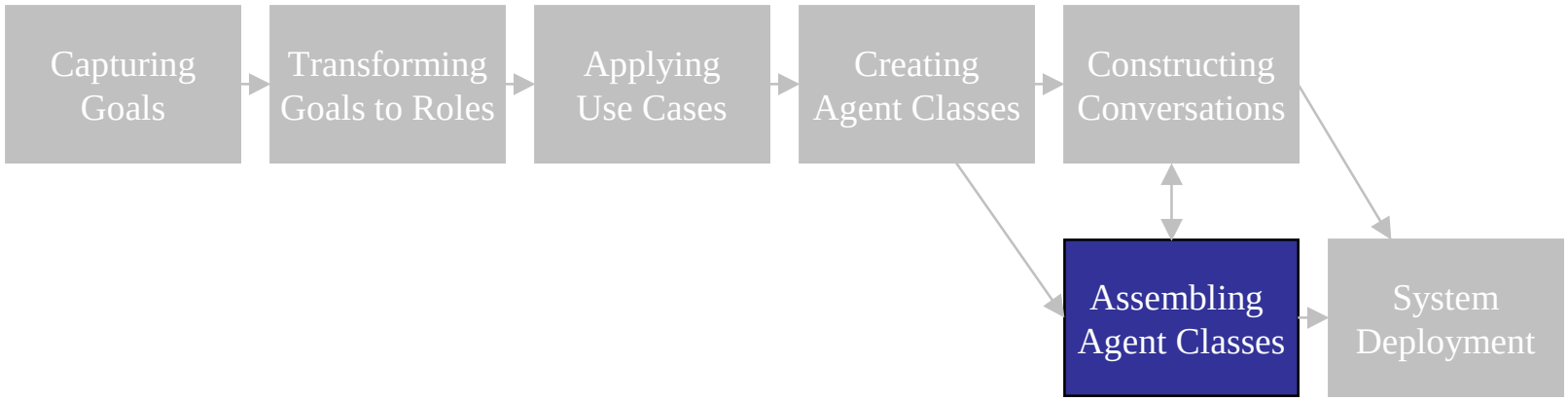
Constructing Conversations



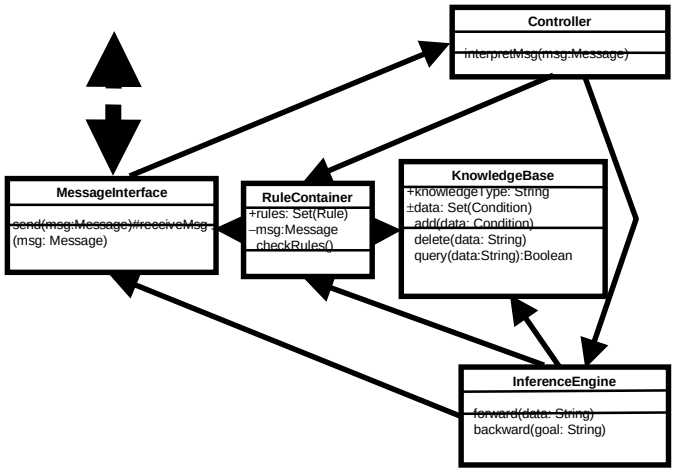
- Input
  - Agent Class Diagram
- Product
  - Conversations as graphical state tables including actions
- Diagrams
  - Conversation Diagrams



Agent Assembly



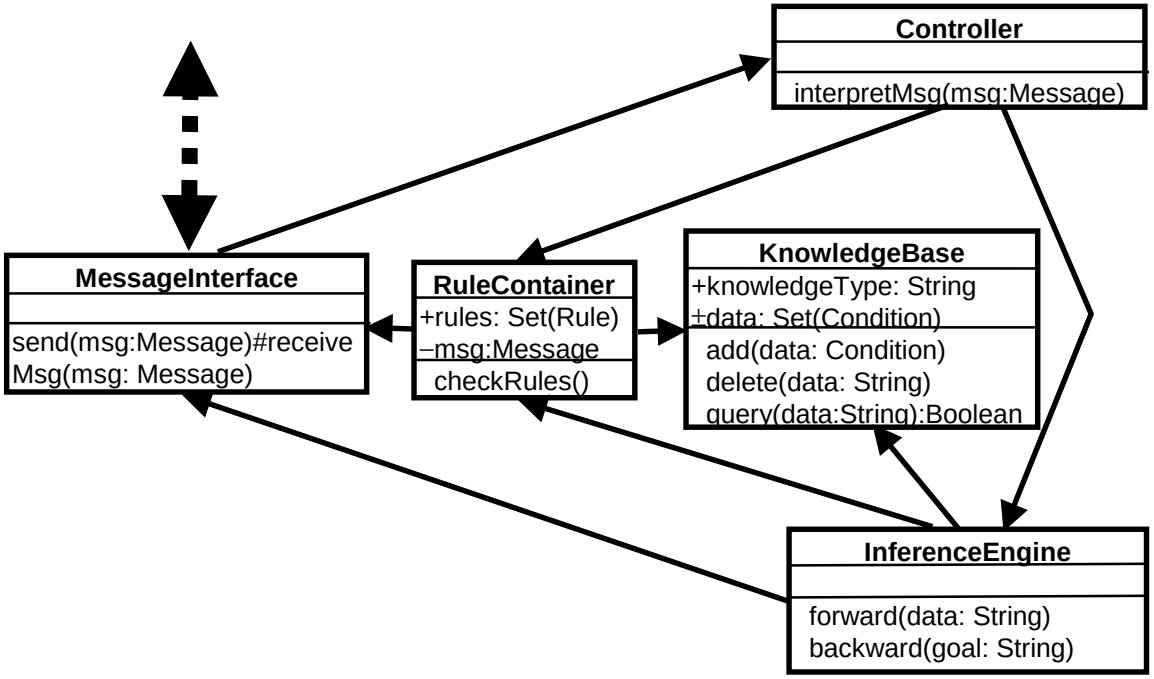
- Input
  - Agent Class Diagram
  - Conversation Diagrams
- Product
  - Complete Agent classes with components
- Diagrams
  - Agent Architecture Diagram



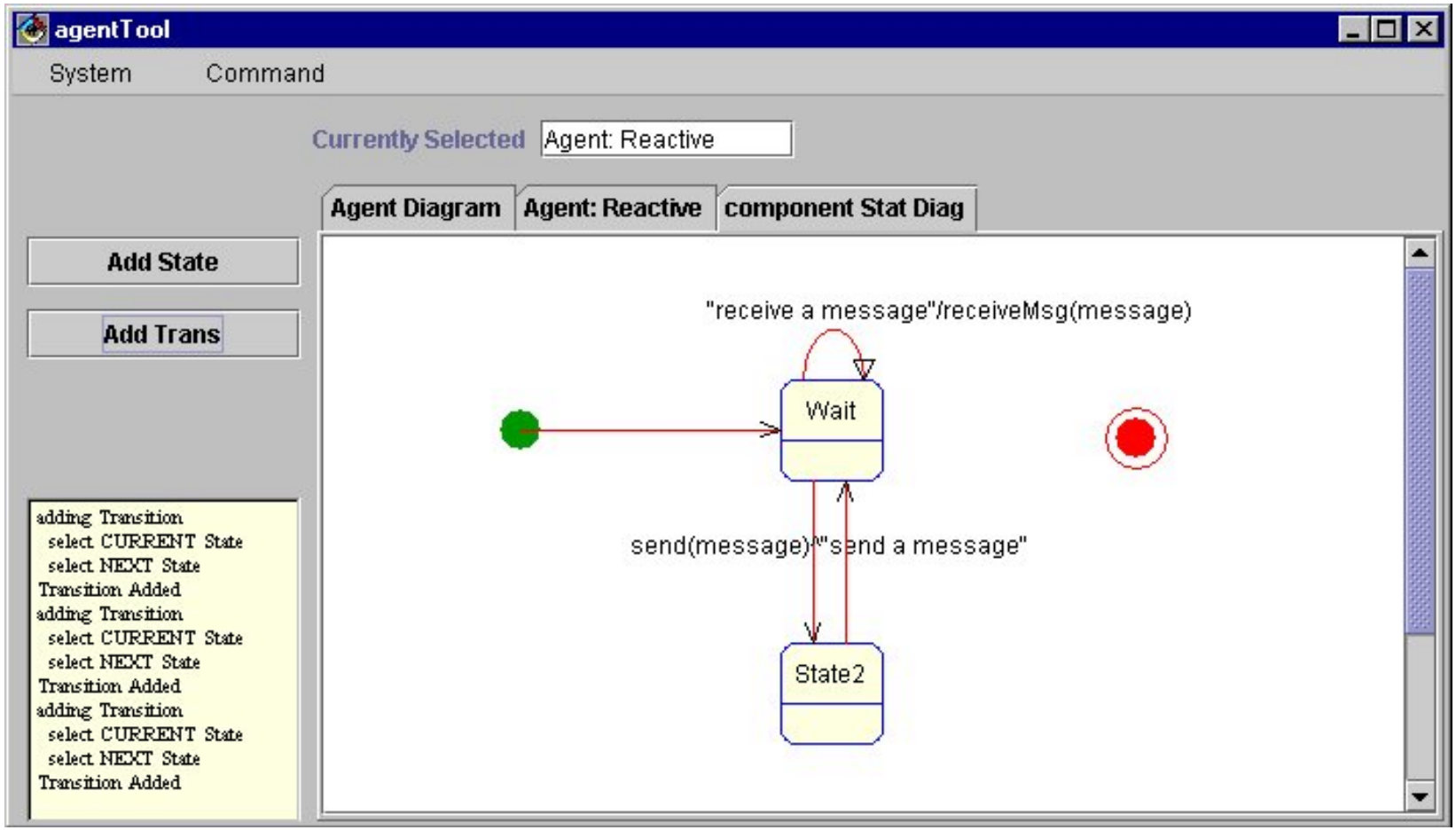
Agent Class Assembly

- Defined using Agent Definition Language (AgDL)
  - Components and connectors
    - depicts static structure of agent
  - Object Constraint Language (OCL)
    - represents low-level definitions
  - State diagrams
    - depicts dynamic aspects

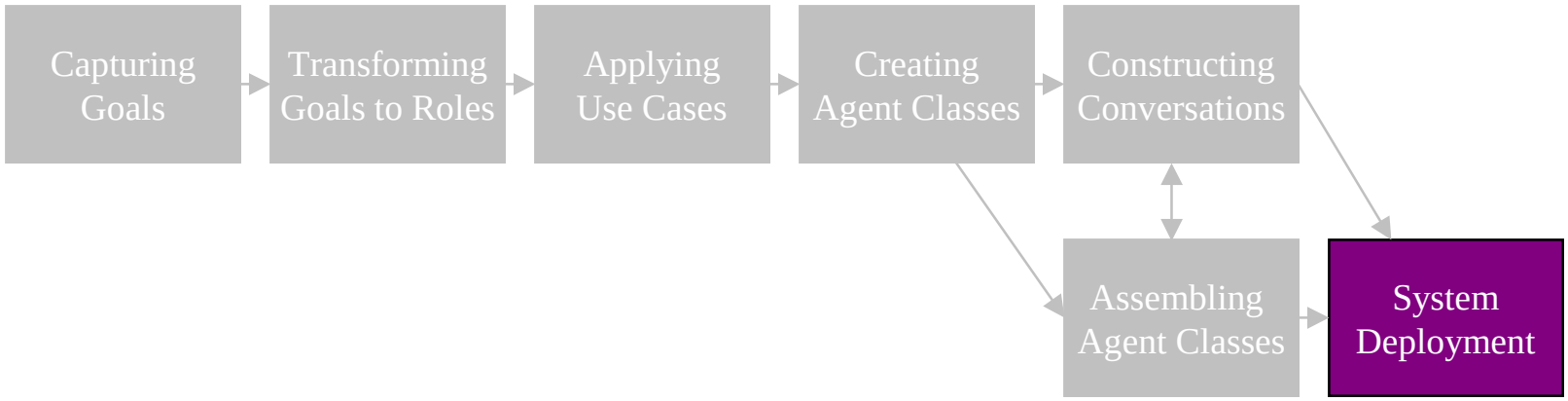
Static Model



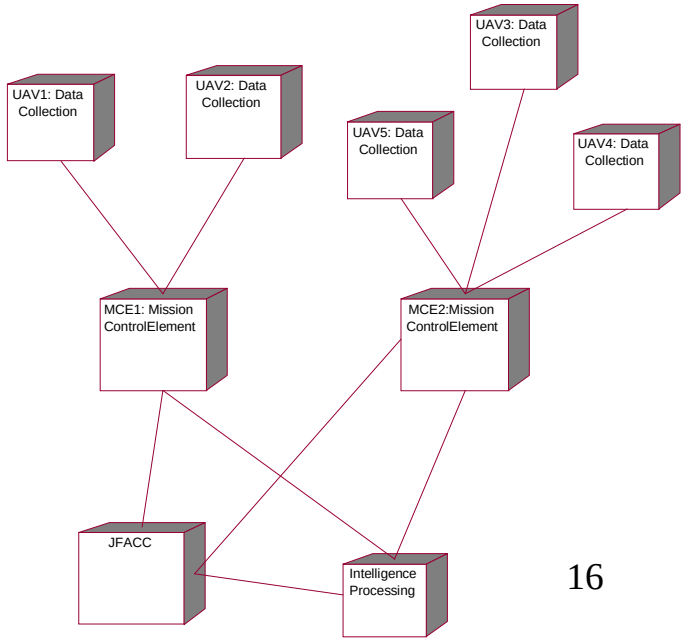
Dynamic Model



Deployment



- Input
  - Complete agent classes
- Product
  - A collection of agents
- Diagrams
  - Deployment Diagram



agentTool

agentTool

Provide users the ability to specify & design agent systems and semi-automatically generate those systems using software synthesis and component reuse techniques.

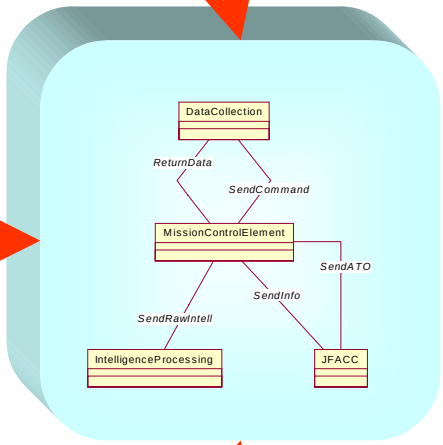
Domain Knowledge

- Multiagent Infrastructures

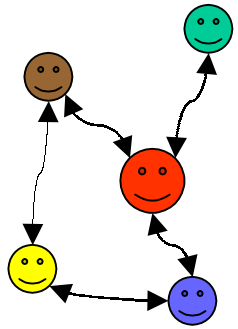
AI Components

- planners
- inference mechanisms
- search algorithms
- learning algorithms

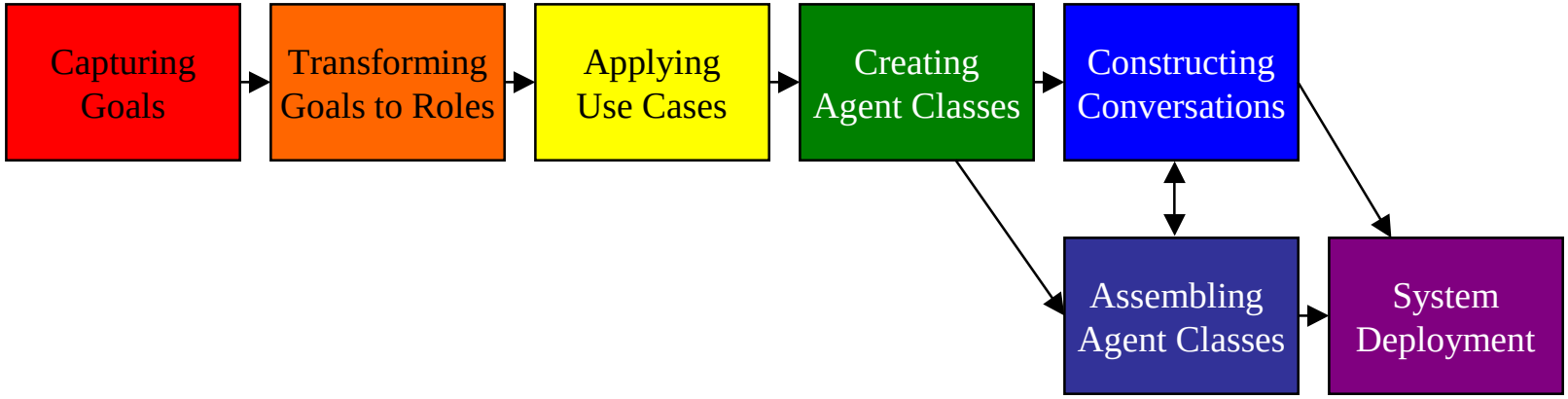
Communication & Security Protocols

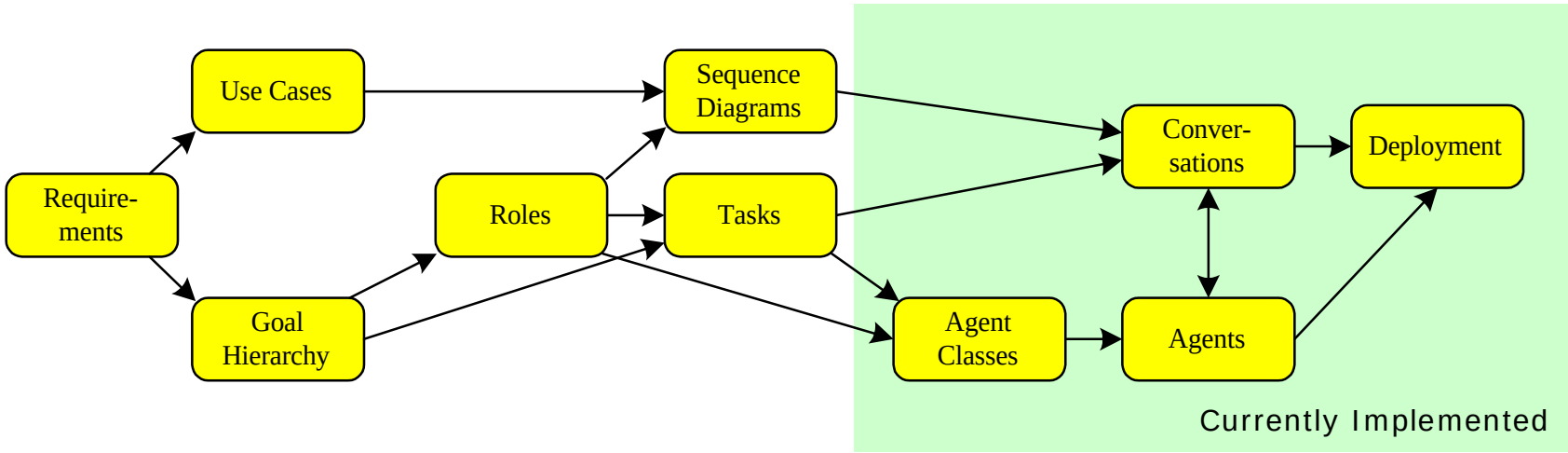


generation

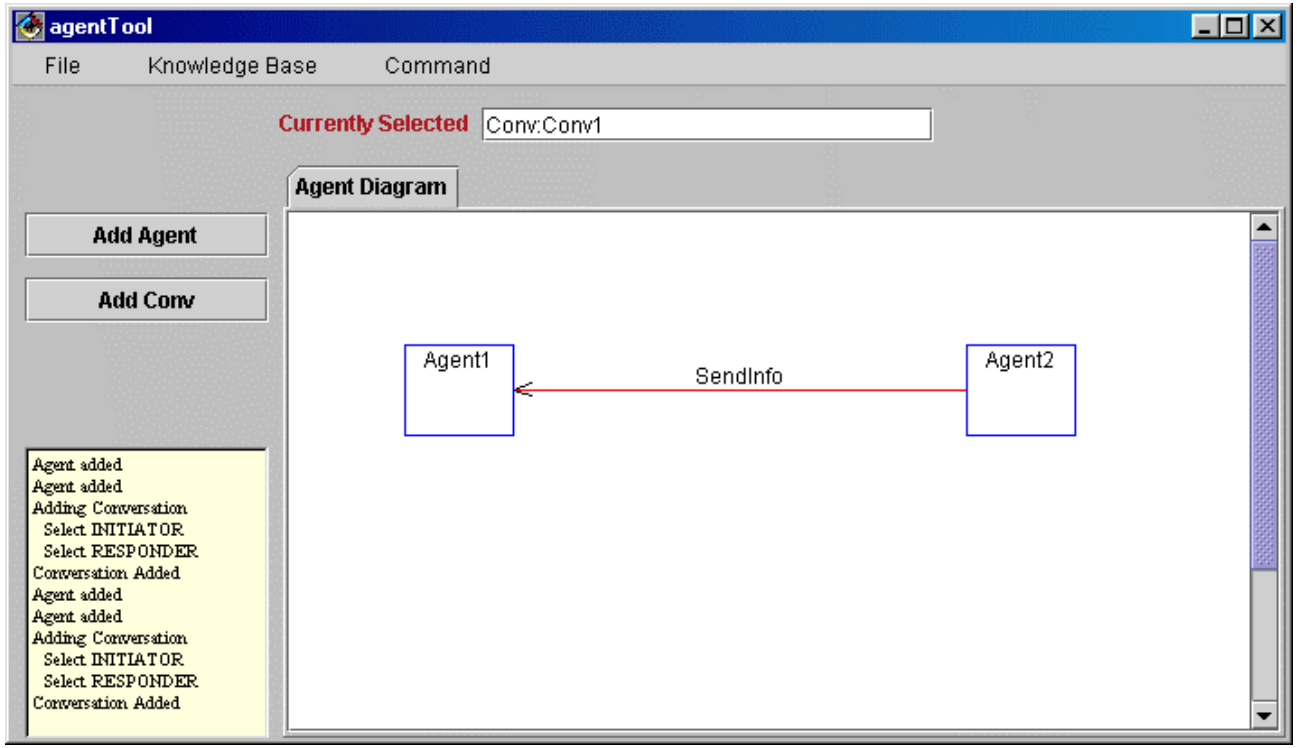


Multiagent Systems Engineering (MaSE)

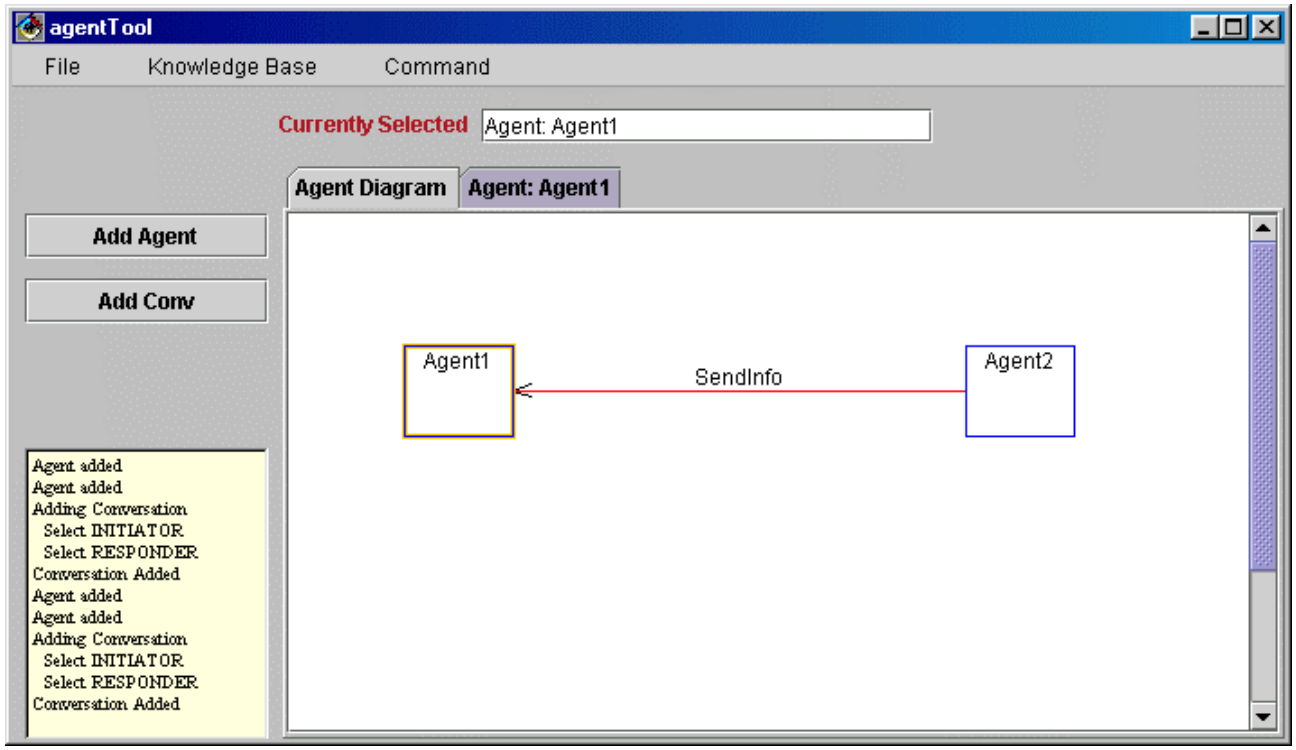




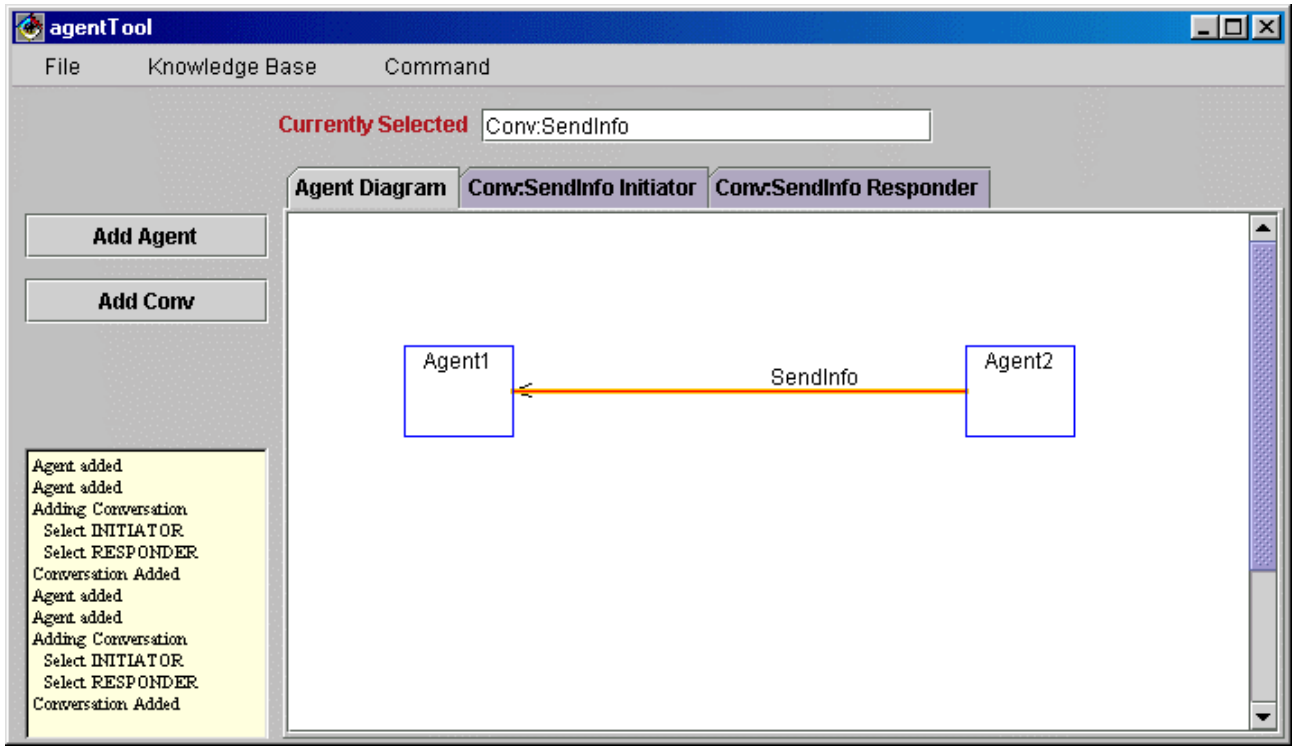
Agent Diagram



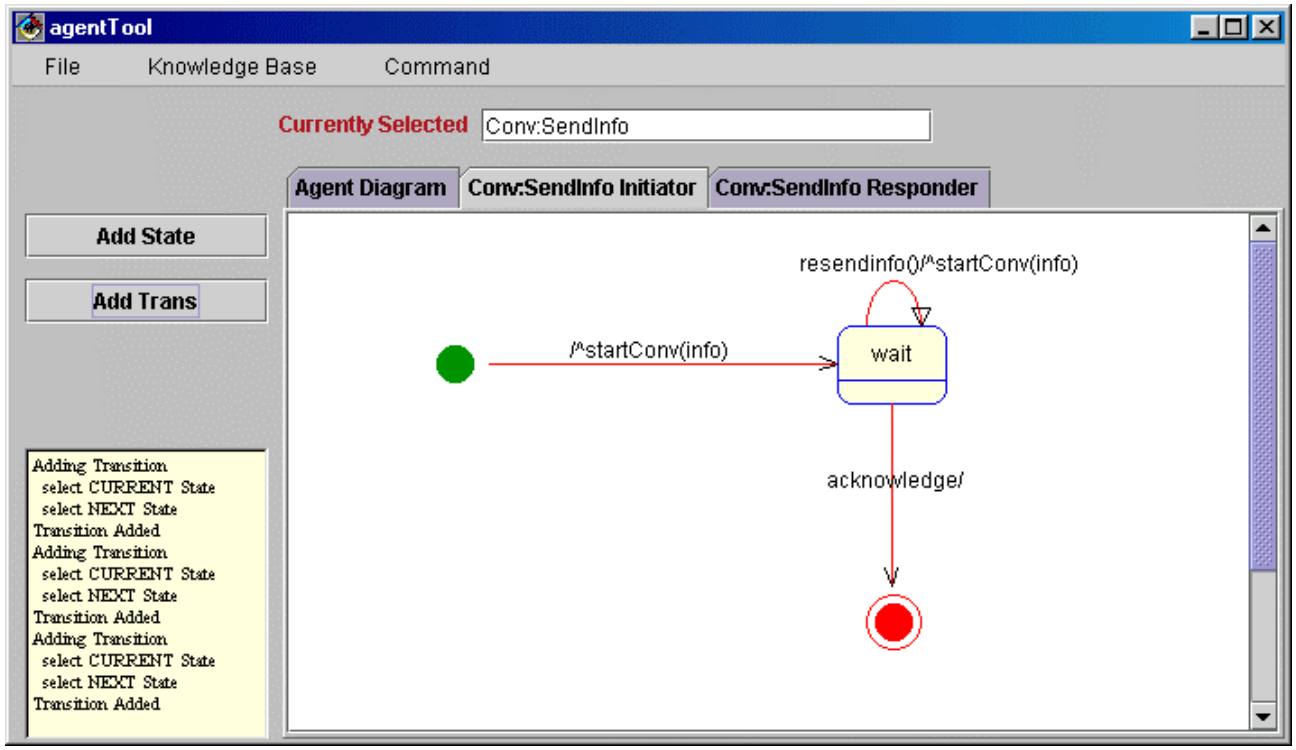
Selecting an Agent Class



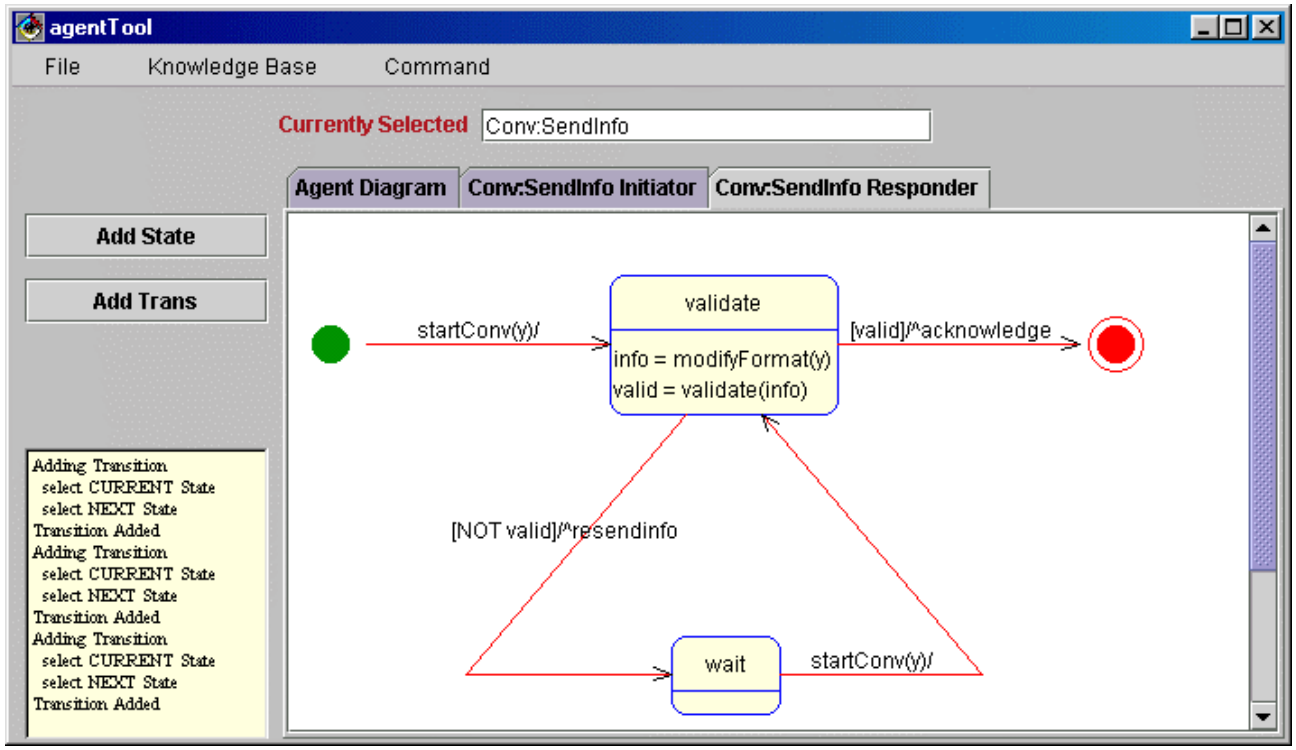
Selecting a Conversation



Conversation Diagram
(half a conversation)

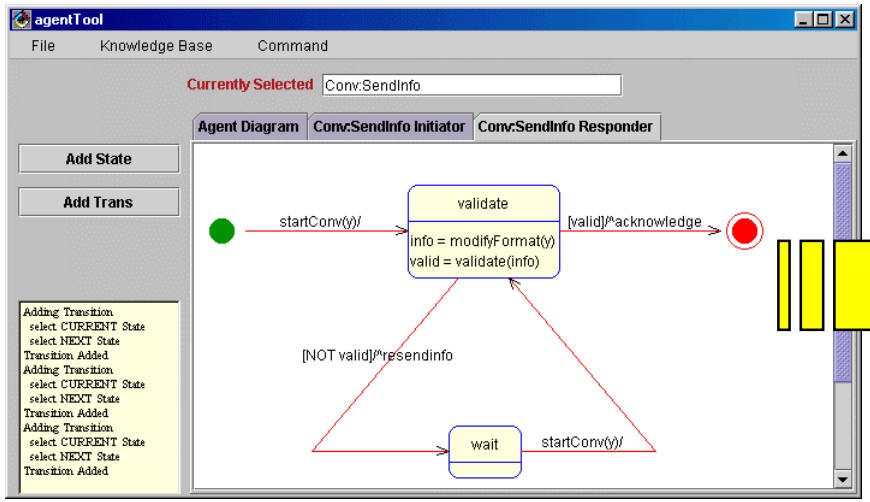


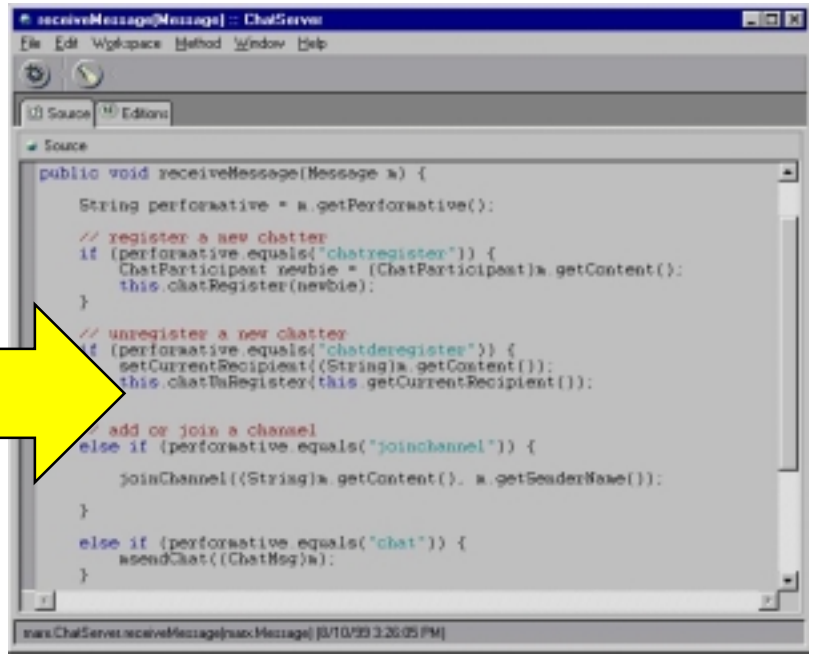
Conversation Diagram
(the other half)



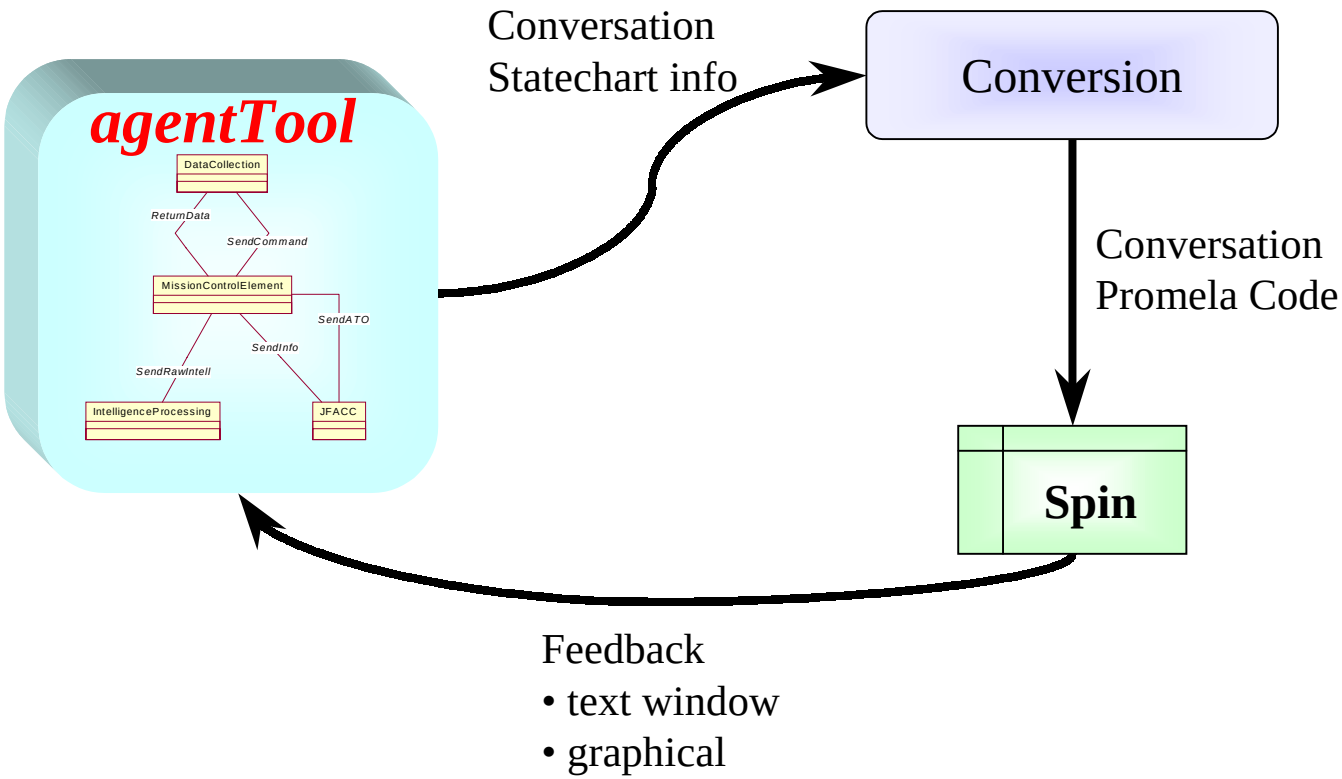
Code Generation

- Automatic from Agent and Conversation Diagrams
- Select platform-dependent components such as a messaging system
- Currently focused on agentMom



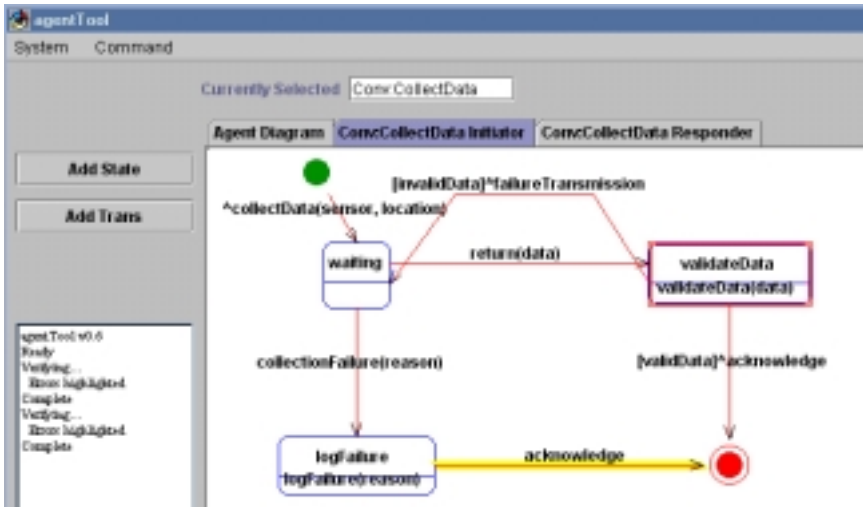


Verification Process



Feedback

```
proc 0 = :init:
proc 1 = SendInfoInitiator
proc 2 = SendInfoResponder
proc 3 = CollectDataInitiator
proc 4 = CollectDataResponder
q\p   0   1   2   3   4
  1   .   .   .   CollectData!collectData
  1   .   .   .   .   CollectData?collectData
  1   .   .   .   .   CollectData!collectionFailure
  1   .   .   .   CollectData?collectionFailure
  2   .   SendInfo!send
  2   .   .   SendInfo?send
  2   .   .   SendInfo!acknowledge
  2   .   SendInfo?acknowledge
spin: trail ends after 16 steps
-------------
final state:
-------------
#processes: 5
 16:           proc  4 (CollectDataResponder) line  92 "verify" (state 27)
proc  3 (CollectDataInitiator) line  65 "verify" (state 24)
proc  2 (SendInfoResponder) line  46 "verify" (state 24) <valid endstate>
proc  1 (SendInfoInitiator) line  25 "verify" (state 22) <valid endstate>
proc  0 (:init:) line 114 "verify" (state 6) <valid endstate>
5 processes created
```



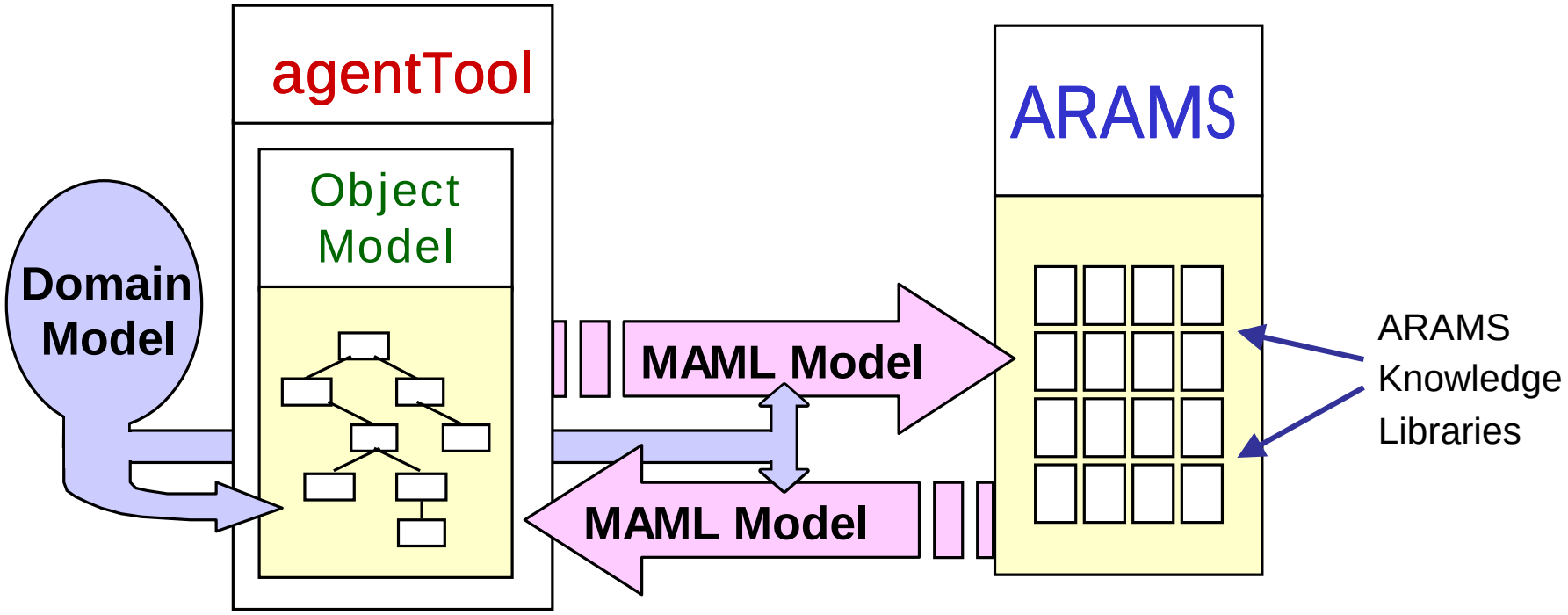

```
DEADLOCK CONDITION EXISTS IN THE FOLLOWING CONVERSATION:
Conversation Name = CollectData
Participant Name = Responder
Current State = wait
State Transition = acknowledge

DEADLOCK CONDITION EXISTS IN THE FOLLOWING CONVERSATION:
Conversation Name = CollectData
Participant Name = Initiator
Current State = logFailure
State Transition = acknowledge
```

Verification Capabilities

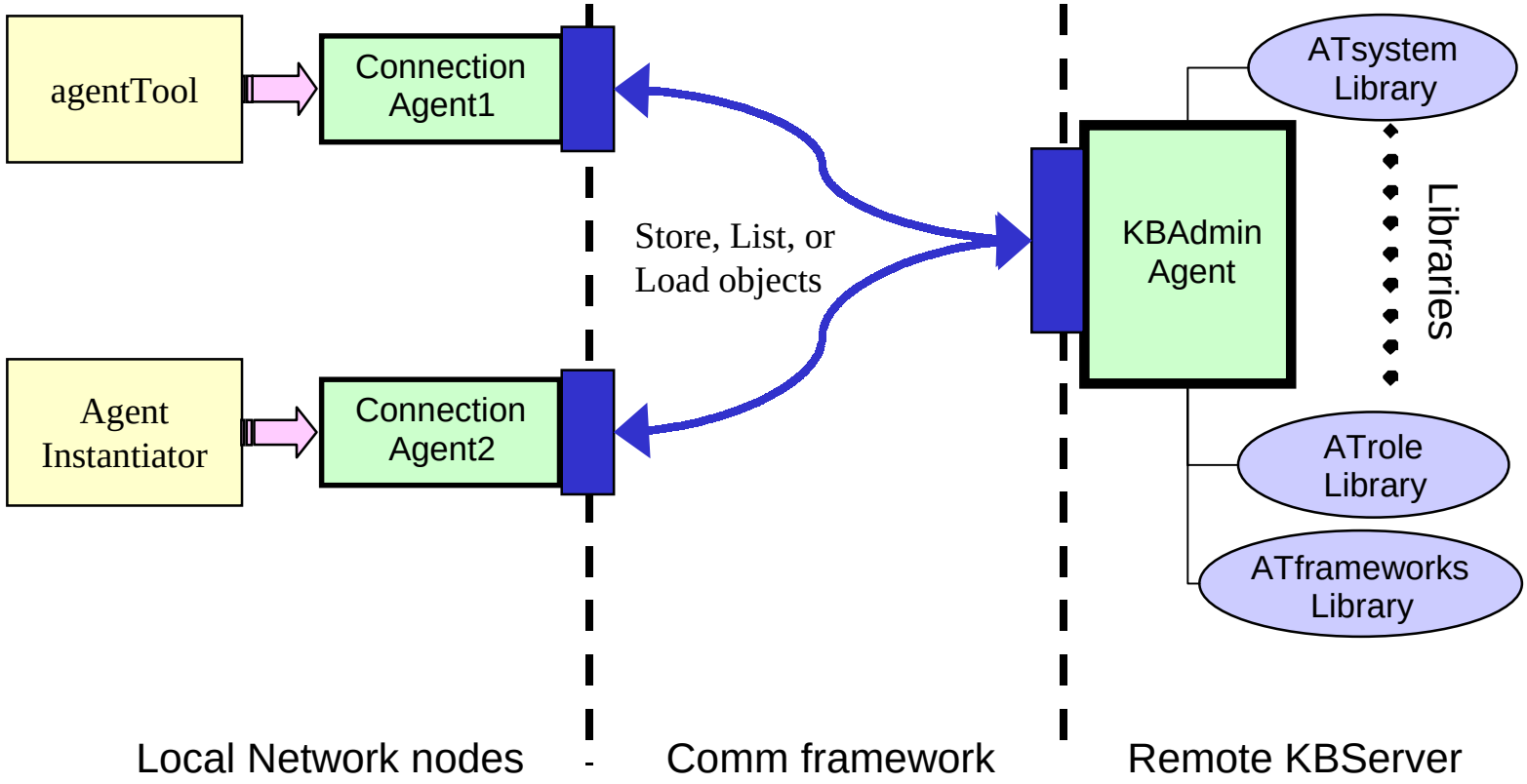
- Conversation deadlocks
- Non-progress loops
- Unused messages
- Mislabeled transitions
- Inability to create required sequences

Knowledge Base Overview

- Agent Random-Access Meta-Structure (ARAMS)
  - Java-based repository
- Multiagent Markup Language (MAML)
  - Key to interoperability – XML based



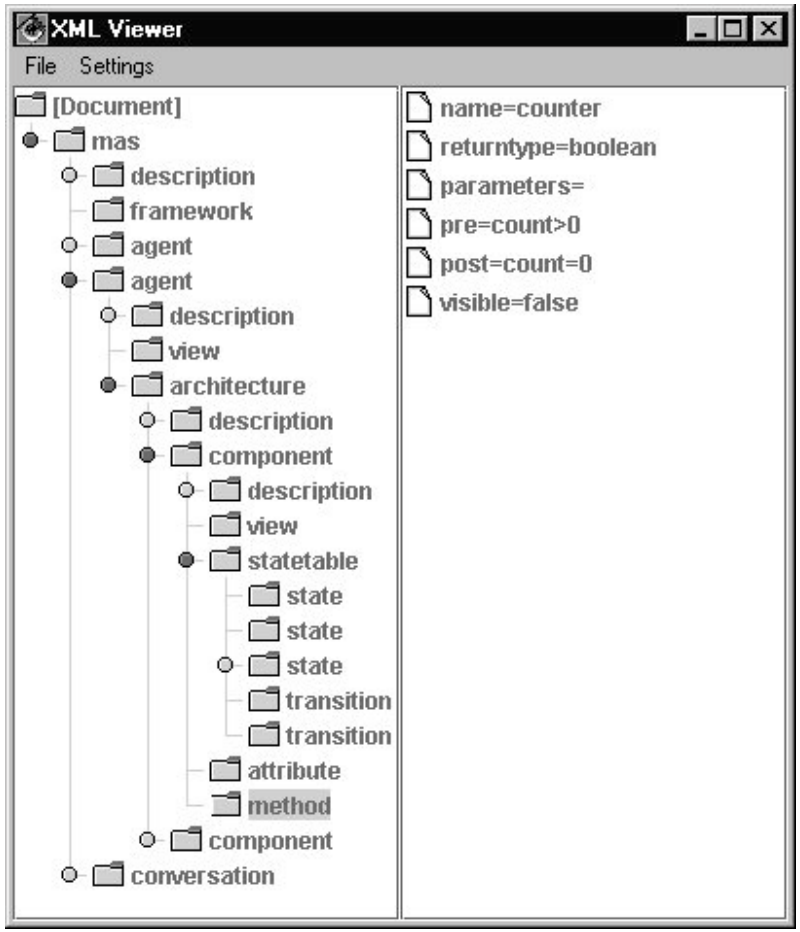
Knowledge Base Implementation



Multi-Agent Meta-Language (MAML)

- **Based on XML**
  - **Compatible with existing tools**
- **Example**

```
</agent>
  <agent version="1" name="Bidder">
     <description>
     none
     </description>
     <view x="421" y="102" w="50" h="50"/>
     <architecture version="1“ name="null">
        <description>
        none
        </description>
        </architecture>
  </agent>
```

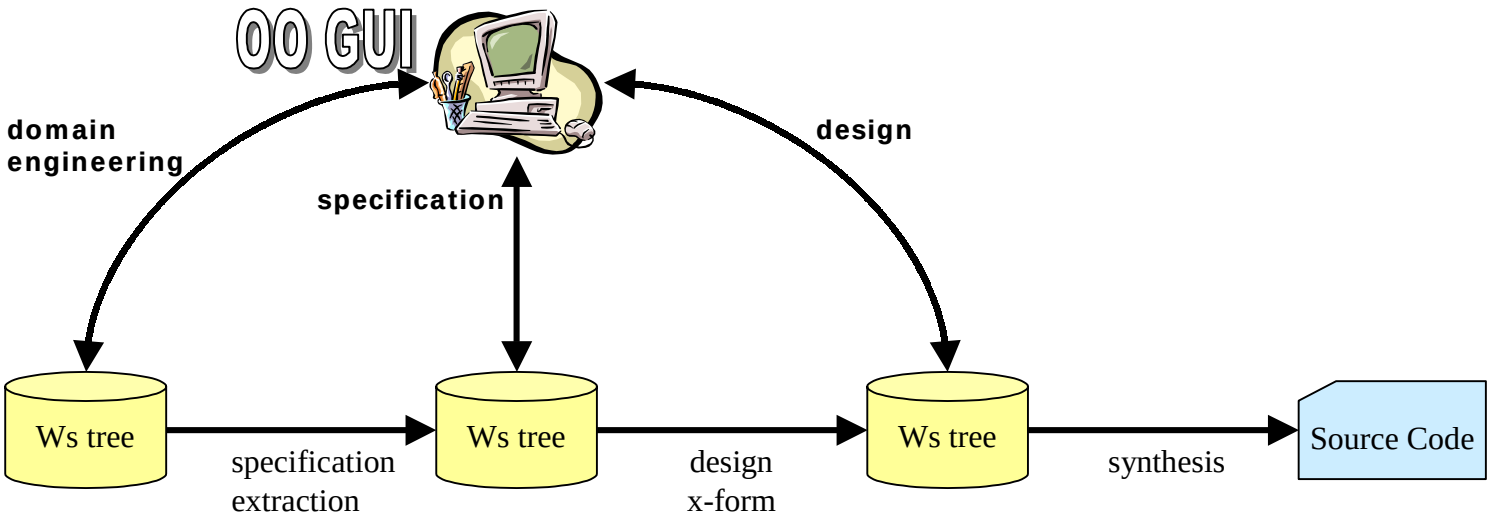


AFIT Wide-Spectrum Object Modeling Environment

(AWSOME)

AWSOME Environment

AFIT Wide-Spectrum Object Modeling Environment



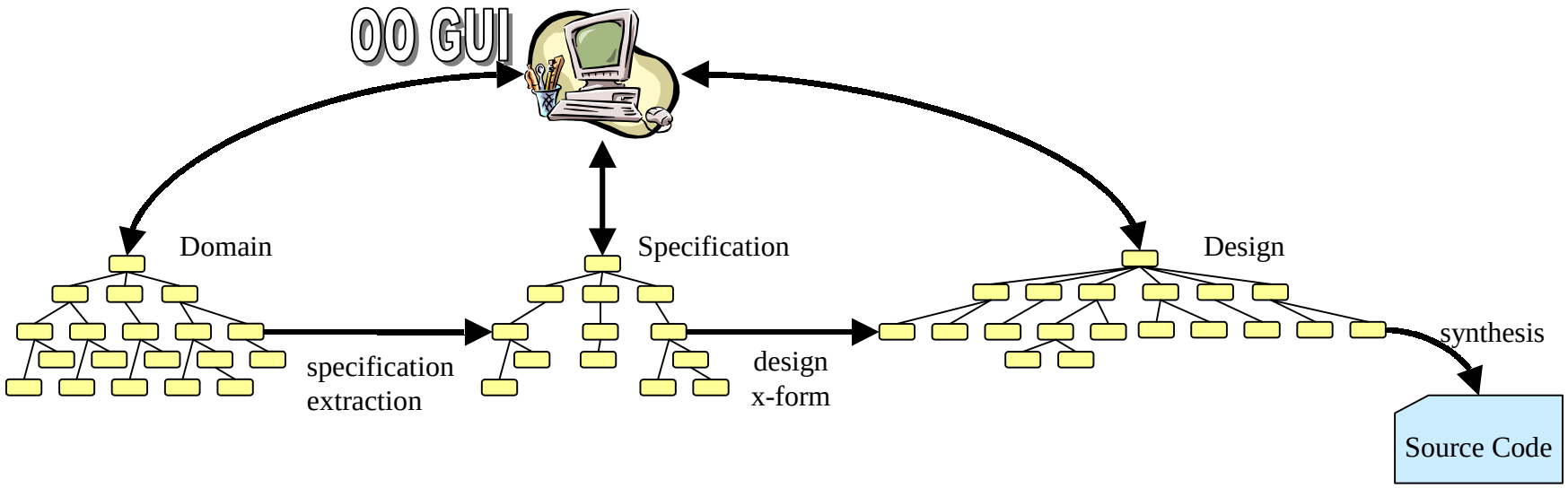
"Domain Tree"

"Specification Tree"

"Design Tree"

AWSOME Wide-Spectrum Modeling Language

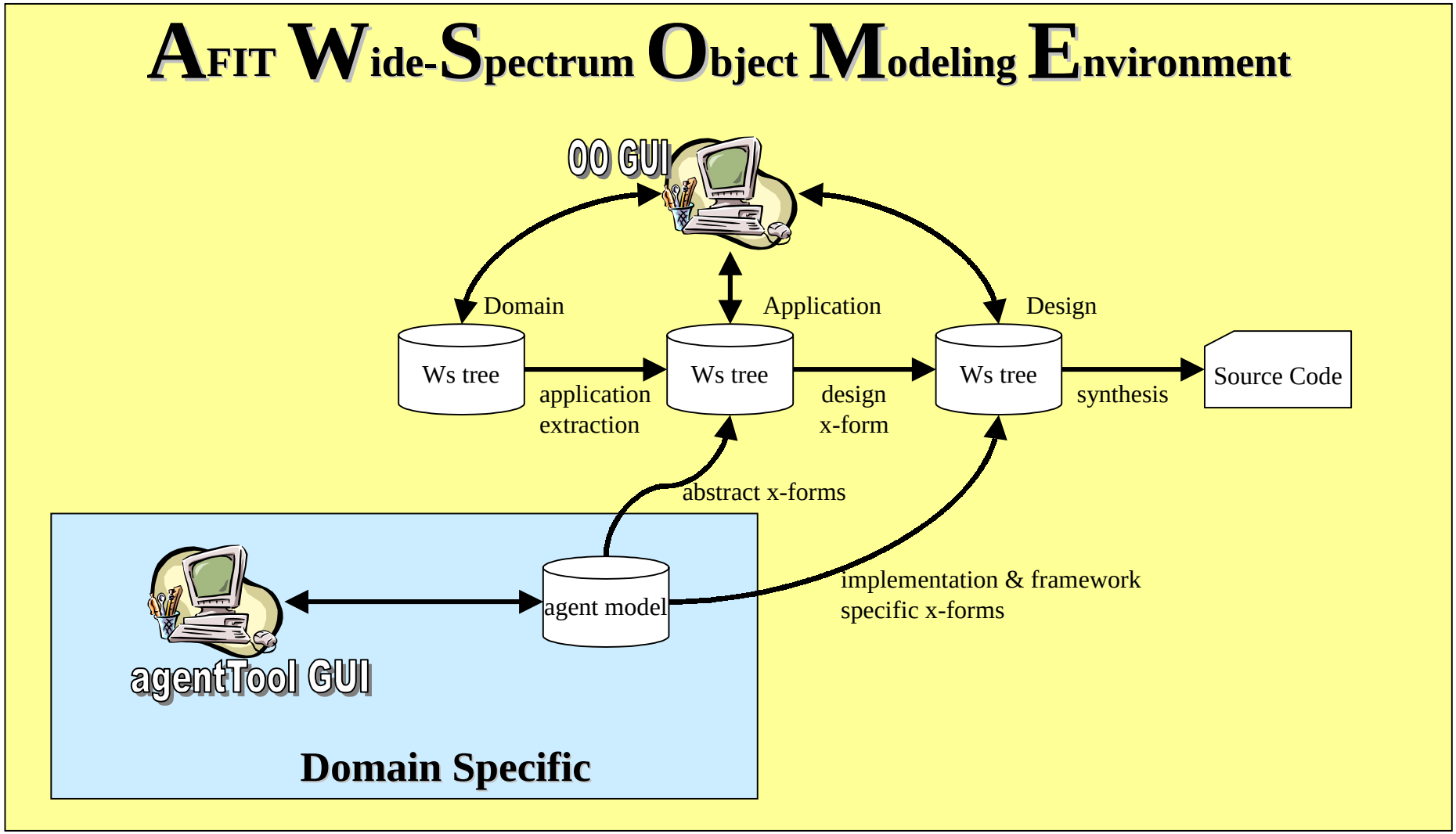
AFIT Wide-Spectrum Object Modeling Environment



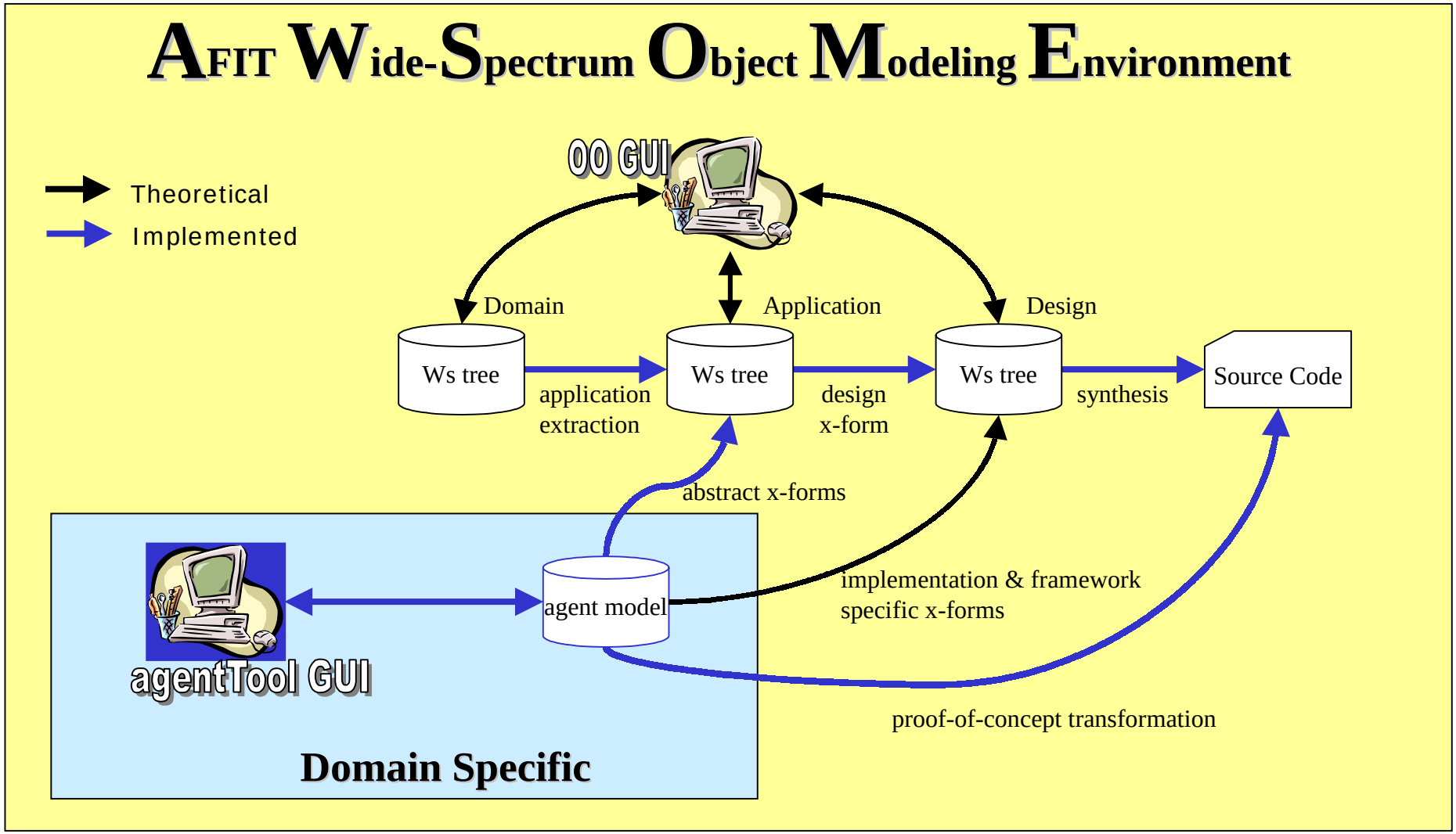
AWSOME Capabilities

- State model defined as formal spec
- Five formal transforms applied to spec
- Resulting design is object model
  - Methods to handle events
  - Method to mimic state transitions
- Java code generated from design model
  - Demonstrated with two agent systems
  - Intend to demonstrate with UConn's MADGS

Domain Specific Front Ends



Domain Specific Front Ends
(Implemented)



Future Direction

- Integration of domains
  - Reuse existing models
  - Communication protocols, security protocols, domain knowledge, etc.
- Detailed design
  - Mapping tasks to conversations and internal agent architectures
- Extend MaSE/agentTool
  - Mobility
  - Dynamic systems

Publications

- Journal – Foundational Work
  - Translating Graphically-Based Object-Oriented Specifications to Formal Specifications. submitted to ACM Transactions on Software Engineering and Methodology, Feb 2000.
  - A Theory-Based Representation for Object-Oriented Domain Models. accepted for publication in IEEE Transactions on Software Engineering, 1999.

Publications

- Conferences & Workshops
  - Developing Multiagent Systems with agentTool, submitted to 7th International Workshop on Agent Theories, Architectures, and Languages, Boston MA, July 2000.
  - An Overview of the Multiagent Systems Engineering Methodology. submitted to 1st International Workshop on Agent-Oriented Software Engineering, Limerick Ireland, June 2000.
  - Automatic Verification of Multiagent Conversations, to be presented at the 11th Annual Midwest Artificial Intelligence and Cognitive Science Conference, Fayetteville Arkansas, July 2000.
  - Design Issues for Mixed-Initiative Agent Systems, AAAI-99 Workshop on Mixed-Initiative Intelligence, Orlando FL, July 1999.
  - Multiagent Systems Engineering: A Methodology and Language for Designing Agent Systems, Agent-Oriented Information Systems '99, Seattle WA, May 99.